Filling the gap between web 2.0 technologies and natural language processing pipelines with semantic web: the semtoolbox approach

P Durville, Fabien Gandon

Filling the gap between web 2.0 technologies and natural language processing pipelines with semantic web

the semtoolbox approach

P. DURVILLE, F. GANDON
EDELWEISS Team
INRIA
Sophia Antipolis, France
{priscille.durville, fabien.gandon}@inria.fr

***Abstract*—As web 2.0 websites are now widespread and natural language proposes mature processing methods, using the later to enrich the former opens new possibilities to improve interactions. This paper will show a way of combining natural language processing tools with semantic web to enhance tagging and mashup functionalities of web 2.0 websites.**

Keywords : Semantic web, tagging, web 2.0, natural language processing.

I. INTRODUCTION

Websites using web 2.0 technologies are now widespread over the web. These websites often include tagging and/or data mashup functionalities into their pages. Semantic web in the broad sense of the term (with RDF [1], RDFa [2], or microformats[1]) is used to publish, to embed and to link data over the web, providing interoperability with a level and a scale unseen before. In web 2.0 applications, data are mainly entered manually and partially computed from dedicated programs such as wrappers.

On the hand, natural language processing (NLP) tools have reach maturity. These standalone applications allow dealing with all kinds of text including, in particular, web pages. Thus there is a clear opportunity to combine the both approaches in order to be able to add some possibilities to websites in terms of tagging suggestions, intelligent scraping and so on.

However the two worlds are currently rarely linked mainly because the technologies used in each approach come from very different culture. While the first one relies on scripting and lightweight components with time performance objectives, the second one relies on standalone heavy applications and is time consuming (applications often run in background).

In this paper, we will see that it is possible to follow an hybrid approach by developing and using lightweight NLP components allowing us to add specific functionalities to web 2.0 applications. In a first part of the paper we will remind characteristics of web 2.0 websites. Afterwards, we will present classical NLP processes that can be performed on texts. In a third part of the paper we discuss how NLP processes can be used to add some goodies to web 2.0 websites. We will show our proposal of a lightweight component relying on semantic web and NLP technologies: Semtoolbox that can be embedded in websites. We will illustrate its use on two examples of portals dedicated to technological monitoring and business intelligent context. We will conclude with work in progress, future work and some perspectives.

[1] http://microformats.org/

II. CHARACTERISTICS OF WEB 2.0 WEBSITES

A. Short definition of web 2.0 websites

"Web 2.0" is widely used to describe a new way of thinking data and data interactions over websites. This evolution concerns technologies as well as practices. In particular, the way users interact with each others by means of acting directly on the websites content (by adding tags to facilitate future search/retrieval/classification for example) shows this evolution. So web 2.0 illustrates community, dynamicity and interactions among the web. Consequently websites rely more and more on tools like wikis, blogs etc. In order to be in accordance with this new perspectives, new functionalities appear like page tagging, social bookmarking, page scraping, micro blogging and so on which allow users to organize and to share data and to easily use resources of others even in another context than the initial one.

B. Some functionalities provided by 2.0 websites

One of the most important functionality provided by 2.0 websites is the ability for users to add keywords on pages (or fragments of pages) to enhance later searches. These keywords can be organized and structured or not. They are composed of one or more words and often describe a concept (belonging to a community's vocabulary and/or shared by a group of users). This makes possible a syntactic or semantic search over a forum or a blog or a blog directory for example. This tagging functionality easily allows pages and resources to be connected with each others. The more a keyword is used over pages, and the more a page tagged with this keyword will be found and seen.

Another functionality relies on content syndication as RSS feeds, IRC channels and collaborative tools. The tagging process is also a way of classifying or organizing them to allow users to create custom interfaces that answer

custom needs. The portals and portlets[2] technologies, for example, are dedicated to the development of such interfaces compounding.

A third point is about web scraping, a mean of extracting useful information from ordinary web pages and reblending them for additional uses in another syntax and/or context (for example putting them in pre-formatted containers such as spreadsheets, XML or databases, with extracted content well organized and semantically accessible).

C. *Limits and constraints*

The functionalities presented above exist and are largely used. But users have to play an important role populating the websites with the appropriate metadata (tags on pages etc), putting together useful information from different sources and so on. Developers play an important role too by creating bridges to translate data from one syntax or context to another. Part of this work is of course necessary. But a significant part could be automated ; this is where NLP can help a lot.

III. NATURAL LANGUAGE PROCESSING : EXISTING FUNCTIONALITIES AND TOOLS

A. *Short description of natural language processing methodology*

NLP tools always rely on a chain of linguistic processes called pipeline. The pipeline can vary from one tool to another but some steps of these processes are classical and essential. First of all, the input must be a plain text. So pre-processing steps are often needed to extract and clean the original text so that it can then be given as an input to the pipeline. Information about the language of the text is crucial, so if we don't know it, a step of language detection is necessary. The next step is tokenization to divide the whole text into tokens (sentences or words for example) that will be processed as linguistic units. Then a step of lemmatization is classically applied to abstract grammatical variations (such as plural forms for example). After that, grammatical identification associates each token with a grammatical category (e.g. noun, adjective, verb and so on). Additional steps can act as filters to eliminate non-meaningful words (like "and", "the" and so called "stop words"). At this stage, next processing steps depend on the objectives followed: extraction of terms, annotation generation, cloud creation and so on.

B. *Existing tools and pipelines*

The tokenization step is not really difficult and there are a lot of existing lightweight components performing this task.

The lemmatization step is a more complex task. Two main approaches can be followed: relying on suffix stripping or relying on languages dedicated grammars. While relying on suffix stripping, each word is truncated with regards to stemming algorithms. An example of such algorithms is the one developed by Martin Porter [3] and distributed within Snowball [4]. A more complex approach to the problem of determining a stem of a word is the one that tries to use the part of speech of a word (e.g. adjective, verb). In this case, each part of speech is used to apply different normalization rules, since for some languages, the stemming rules change depending on a word's part of speech. An example of such approaches is the Durm German Lemmatizer [5].

The word category disambiguation can be done programmatically by using both probabilistic methods and manually tagged training corpus. An example of this method and associated tool is TreeTagger [6].

At last, filters steps, like tokenization step, are easy to develop. They always rely on words and expressions lists and/or regular expressions. The difficulty here is to have such a relevant list dedicated to our purpose.

In order to process a whole pipeline throughout these steps, it can be useful to delegate the tools combination and integration mechanism and the data flow management between them to a framework. Well known frameworks like Gate [7] or UIMA[3] are dedicated to this architecture task.

C. *Limits and constraints for embedded use in websites*

As we have seen, a whole linguistic pipeline requires various tools that can be heavy components requiring long time execution, system dependant components that cannot be deployed everywhere neither embedded easily in a web application. Moreover, in order to combine the different components in a single pipeline and make them interact between each others, it is necessary to use a global framework dedicated to that purpose. These frameworks have a lot of dependencies that unluckily could be in conflict with other components.

Some "on the shelf" applications exist but they are distributed as rich clients with a dedicated graphical interface and are not designed to be connected with another application. When they happen to be accessible on the web, it is usually not as services but only as manual web applications, like the Semato [4] project for example. Moreover, these applications are mainly dedicated to a specific domain application or a specific end-user need. For example, the ProxiDocs [8] tool is a standalone tool that will display maps describing a corpus of texts. And there is no possibility to get the results in a given exchange syntax to use them as input in another tool (a website for example).

IV. FILLING THE GAP BETWEEN THIS TWO WORLDS

In this context, lightweight libraries dedicated to one specific task (tagging, terms extraction, and so on) and relying on simplest components would be really useful.

A. *Needs and functionalities of a lightweight library*

As we have seen above, tagging web pages is now a classical task in web 2.0 websites. This tagging process would be enhanced and optimized if suggestions are displayed to users or if tags are added by default programmatically according to page content and/or page context. Users may then be able to modify or add

[2] http://jcp.org/en/jsr/detail?id=168

[3] http://incubator.apache.org/uima/index.html

[4] http://semato.uqam.ca/guidexpert-ato/gea.asp

supplementary tags. This may be applied to IRC channels or RSS feeds too to generate short views displayable on dedicated portals or specific media. While adding a new bookmark in a web browser, generated tags can be used to sort it into the bookmarks tree. An identical process may be applied to emails in webmail editors. For this kind of tagging processes, a lightweight easily embedded library is required. An example of such library, our library Semtoolbox, is described in the next section.

B. An example of such a library : Semtoolbox

The semantic web aims to make it possible to share data on the web and make them accessible for both machine and users. It provides languages (e.g. OWL [9]) to describe data meaning. Using these languages, domains are described as ontologies containing domain concepts and properties. These concepts allow categorizing resources, and the properties describe links between them. All these information can be kept as RDF annotations. This technology fits the tagging process need of getting tags meaning. Our library takes advantage of this to provide some tagging functionalities.

Semtoolbox is a lightweight library developed in Java. It allows developers to find concepts of one (or more) specified OWL ontology(ies) inside texts. The matching process between text and concepts relies on concepts' labels. It can be strict, case insensitive or "grammar insensitive". For that last kind of matching, the snowball stemmers are used. The same functionalities are provided for properties of one (or more) specified ontology(ies) and, in a more general way, for every kind of semantic resources available in a knowledge base (here a set of ontologies, annotations and rules accessible locally or on the web). In that last case when using any kind of resources, in order to create an index containing all the resources to look for in the text, and to have no constraint of what kind of resources it is, we use a SPARQL query [10] to denote them. Thus, resources can be values of properties or much more complex computing resources. This SPARQL query is sent to a semantic engine, CORESE [11], which will return a SPARQL result used to create a dedicated and temporary index.

When no ontology neither knowledge base is provided, the library acts as a term extractor. The default index used in that case to look for terms in a text is built using the WordNet[5] lexical database. This lexical base is used to filter words and reduce the noise generated otherwise. The tagging process can be viewed, in this case, as the first step of an ontology creation process. This first step deals with the creation of a draft vocabulary that can be later refined, completed, organized and structured.

Results of this tagging process is provided as follows:

- a list of found resources (semantic resources or WordNet terms),
- a list of found resources and, for each resource, its frequency into the text,
- a list of found resources and, for each resource, the exact word(s) or expression found into the text, the start and end offsets of that word(s)/expression into the text. This can be useful to highlight words in a text editor for example.

As is, the Semtoolbox library does not exceed 40Ko and, as it is written in Java, it can be embedded in J2EE web applications. A web service version of the library has also been developed so that web applications (in every language) can call Semtoolbox tagging functionalities in a SOAP or REST way.

C. Examples of use of the Semtoolbox library in a web 2.0 context

One of the aims of the e-WOK_HUB[6] project is to develop a custom portal dedicated to CO_2 storage project memory and technological monitoring. Users are geologists and domain experts. They will use the portal in a bibliographical way to look for papers, reports, projects and so on dealing with given geological periods and/or geographical areas and/or geological objects.

Each time a new document comes into the database, it is automatically processed by a chain of web services. One of these services generates RDF annotations on the document that will be stored in a knowledge base and used by search processes. These annotations contain information about the geographical areas of interest the document deals with, the geological period(s) concerned by these areas and the geological objects considered inside these areas. In order to generate these complex annotations, a first step of text indexation is performed using the Semtoolbox library. This step generates tags on the text. The tags represent either:

- concepts coming from the dedicated geological ontologies (example : Limestone or Chalk coming from an ontology dedicated to lithology domain),
- instances of known geological objects,
- instances coming from geological dating RDF base,
- instances of areas defined in the COG[7] (extended with geological projects areas) RDF base.

The annotation generation process relies on these tags and on other linguistic aspects to produce RDF annotations.

While looking for interesting documents, an expert select one of the found documents in order to visualize its content and the associated tags. Each term in the text, source of a generated tag, is highlighted in a different color relative to the ontological origin of the tag. Statistical information about these tags are displayed to the expert so that he has a global view of the document itself and the document regarding the other documents found in the same search request.

The other example of possible use of the Semtoolbox library takes place in the ISICIL project[8] dedicated to web 2.0 semantic portals supporting technological monitoring and business intelligence experts in their job. These portals rely on web 2.0 advanced interfaces (blog, wiki, social bookmarking) for interactions and on semantic web technologies for interoperability and information processing.

[5] http://wordnet.princeton.edu/

[6] http://www.inria.fr/sophia/edelweiss/projects/ewok

[7] http://rdf.insee.fr/geo/

[8] http://isicil.inria.fr/

In these composed portals, the users need a tool to visualize evolution of the dedicated domain, new trends and so on. The final objective of the portal is to visualize hot-lists of tags on given themes and to be able to track the associated experts within dedicated websites. In this context, the Semtoolbox library can be used to extract terms from textual documents in order to help experts to enrich their domain ontologies while the domain evolves. The Semtoolbox library can also be useful in a second step to extract concepts of these ontologies from textual resources. That tagging process allows the portal to display clouds of tags and networks of tags. The idea is to provide users with graphical tools embedded in the portal to compare the tags clouds obtained from different sources.

V. CONCLUSION AND PERSPECTIVES

As web 2.0 websites are now widespread on the web and require more and more data and usage interoperability, the need for lightweight tools that can be easily embedded in web applications to enhance user experience and data management by users is increasing. Natural language processing methods can help us answer this need but the existing linguistic tools implementing the NLP methodology are often rich clients dedicated to standalone and non-collaborative use. The linguistic processes are time consuming and dedicated to run on large corpus while lightweight tools needed by websites do not fit these aspects. Websites often need to process small texts or parts of text and the response time must be immediate. These constraints can be addressed by developing partial linguistic pipelines dedicated to one specific task.

Automated tagging process is typically one of the tasks that can be carried out by such lightweight components. We have seen that tagging process is useful in many contexts like wikis, blogs, IRC channels, RSS feeds and so on. This tagging process can rely on dedicated predefined indexes (from semantic knowledge base for example) or from scratch (acting as a terms extractor).

Therefore, we proposed Semtoolbox, a lightweight library providing tagging functionality. In this paper we presented two examples of use of this library in portals dedicated to technological monitoring in different application domains. While implementation and tests have been carried out in the context of the e-WOK_HUB project, the ISICIL project is at its beginning stage. Consequently, new functionalities and use-cases are still coming up to make the library evolve. In this second context, we hope to have opportunities to work not only on textual documents (PDF, .doc and so on) and web pages but also on alternative data such as RSS feeds, IRC channels and emails.

One of the perspectives for our work deals with terms extraction without predefined indexes. For now, we rely on WordNet to filter words or expressions found in the text after “stop words” removal. As we are also interested in terms containing two or three words, an interesting add-on will be to consider combination of found terms. But in order to avoid redundancies and non meaningful terms, we have to filter the combination by removing the ones included in another one already found in the text for example.

As the Semtoolbox library relies on knowledge base (or upon Wordnet) to build its index, we do not work only with a flat list of terms but also underlying graphs. Consequently, a future work would be to take advantage of this additional information to determine tags correlation or to bring together some found tags and get structured or connected tags networks on textual resources.

Another perspective of this work is to develop other functionalities apart from tagging process. In particular in a web scraping context, there are some goodies that can be developed using lightweight natural language processing components.

Finally, a set of lightweight graphical web components displaying results of the Semtoolbox library would be great to have, like generic text editor with terms highlight functionality or tags clouds views for instance.

ACKNOWLEDGEMENT

We thank the ANR for funding the e-WOK_HUB project ANR-05-RNTL-02702 and the ISICIL project ANR-08-CORD-011 that led to these results.

REFERENCES

[1] F. Manola, E. Miller, B. McBride, RDF primer. Technical report, W3C Recommendation (2004) w3.org/TR/2004/REC-rdf-primer-20040210/.

[2] B. ADIDA, M. Birbeck, RDFa Pimer. Technical report, W3C Recommendation (2008) w3.org/TR/2008/NOTE-xhtml-rdfa-primer-20081014/

[3] M.F. Porter, An algorithm for suffix stripping. *Program*, 1980, vol. 14, pp. 130-137.

[4] M.F. Porter, Snowball: a language for stemming algorithms, 2001, available at: www.*snowball*.tartarus.org/texts/introduction.html

[5] P. Perera, R. Witte, A Self-Learning Context-Aware Lemmatizer for German, Proc. Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP 2005), Association for Computational Linguistics, Oct. 2005, pp. 636-643.

[6] H. Schmid. Probabilistic part-of-speech tagging using decision trees. Proc. International Conference on New Methods in Language Processing, 1994.

[7] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, Gate: a framework and graphical development environment for robust NLP tools and applications. Proc. of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL’02), Philadelphia, USA (July 2002)

[8] T. ROY, P. BEUST, ProxiDocs, un outil de cartographie et de catégorisation thématique de corpus, 7èmes Journées internationnales de l'Analyse statistiques des Données Textuelles (JADT 2004), pp.978-987, Louvain-la-Neuve (Belgique), Mars 2004.

[9] D. L. McGuinness, F. van Harmelen, OWL Web Ontology Language Overview, Technical report, W3C recommendation (2004) http://www.w3.org/TR/owl-features/

[10] E. Prud’hommeaux, A. Seaborne, Sparql query language for rdf, Technical report, W3C Recommendation (2008) www.w3.org/TR/rdf-sparlq-query/.

[11] O. Corby, R. Dieng-Kuntz., C. Faron-Zucker, Querying the semantic web with the corese search engine. Proc. of the 16th Eur. Conf. on Artificial Intelligence ECAI’04/PAIS’04, Valencia, Spain, IOS Press (2004) 705–709.